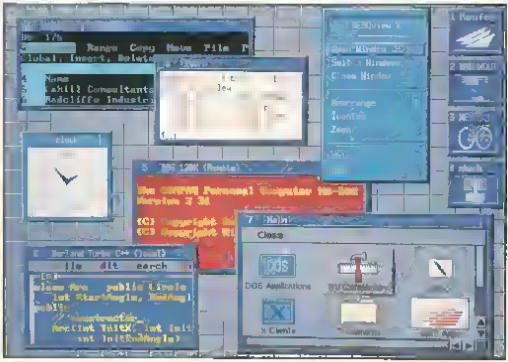
DESQview/X™

Extending DOS into the 21st Century



A Technical Perspective

Contents

1 Executive Summary and Introduction 1

2 DESQview and PC Fundamentals 3

2.1 Introduction 3
2.2 The PC Memory Map 3
2.2.1 Conventional Memory 3
2.2.2 Extended Memory 4
2.2.3 Expanded Memory 4
2.3 EMS 3.2 4
2.4 EEMS and EMS 4.0 4
2.5 Software Emulation of EMS memory 4
2.6 Multitasking more than 640K 4
2.7 LOADHI capability 5
2.8 Program Swapping 5
2.9 Switching and Windowing 5
2.10 Application Video Behavior 5
2.10.1 Well-Behaved Applications 6
2.10.2 Ill Behaved Applications 6
2.10.3 Virtualization 6
2.10.4 Loaders 6
2.11 Application Types 6
2.12 DESQview API 7
2.12.1 The SHADOW Call 7
2.13 Processor Types and Modes 7
2.13.1 The 8088/8086 and Real Mode 7
2.13.2 The 286 and Protected Mode 7
2.13.3 The 386 and V8086 Mode 8
2.13.4 The 486 8
2.14 DOS Extenders 8
2.15 VCPI Specification 8
2.16 DESQview Capabilities 9

3 An Introduction to X 10

3.1 Traditional Graphics Output 10
3.2 X Servers Clients and Protocol 10
3.3 An Event-Driven System 10
3.4 A Distributed System 11
3.5 Operating System and Architecture Independence 11
3.6 X Terminals 11
3.7 A Stand Alone System 12
3.8 The Window Manager 12
3.9 X Development Layers 12
3.9.1 Xlib 13
3.9.2 Toolkits 13
3.9.3 Intrinsics and Widgets 16
3.9.4 Xt Intrinsics 17
3.10 Xt1 17
3.11 Toolkit Summary 17

4 DESQview/X 18

4.1 General Structure 18
4.2 The X Server 19
4.3 Standard DOS Applications 19
4.4 Standard DOS Extended Applications 19
4.5 Standard DOS Graphical Applications 19
4.6 X Clients 19
4.7 The Window Manager 20
4.8 The Network Connection 20
4.8.1 Remote Shell 21
4.8.2 Remote Clients 21
4.9 Stand Alone or Networked? 22
4.9.1 Unix Machines and DOS programs 22
4.9.2 DOS Machines and Unix Programs 22
4.9.3 16-bit DOS Machines and 32-bit DOS Programs 22
4.10 A User's View 23
4.11 A Consistent Growth Path 23
4.11.1 DESQview 23
4.11.2 DESQview 386 24
4.11.3 DESQview/X and DWM 24
4.11.4 DESQview/X and OSF/Motif OPEN LOOK Window Managers 24
4.12 DESQview/X System Capabilities 24
4.13 System Configuration 25
4.14 Summary of DESQview/X 25

5 Development Issues 26

5.1 Real Mode Applications Development 26
5.2 Protected Mode Applications Development 26
5.2.1 Using a Protected Mode Compiler and Linker 26
5.2.2 Using a Standard DOS Compiler 27
5.3 DOS Extenders 27
5.4 X Client Development for DESQview/X 28
5.5 X Program Libraries 28

6 DESQview/X Products 29

6.1 The DESQview/X End User Product 29
6.2 The DESQview/X Development Products 29
6.3 The Network Software Product 29

1 Executive Summary and Introduction

In 1991 an exciting new product from Quarterdeck Office Systems will become available. DESQview/X. DESQview/X a third DESQview in the DESQview suite of multitasking extends the power of PCs by bringing the complete X Window System to DOS.

This booklet is intended to provide end users and developers with an understanding of the capabilities and implementation of this new product along with the DESQview DOS multitasker behind the X Window System.

Definitions

X Window System

The X Window System (also commonly referred to as X11 or X) supports high-performance graphics standard for graphical user interfaces. Developed at the Massachusetts Institute of Technology in 1984, the X Window System has become a de facto industry standard employed by many vendors such as AT&T, DEC, Hewlett-Packard, IBM, Sun Microsystems, and others.

X Server

An X server is a special X Window System graphics application that controls the input and display screen drawing of windows, text, lines, pictures, circles, polygons, and the like according to the requests (messages) from an application. An X Server can handle all the screen drawings for multiple applications concurrently, each application typically displaying information within one or more windows on the screen.

X Client

An X Window System application program (such as a spreadsheet or drawing program) that communicates with an X Server is called an X Client. A well-defined messaging system links the two participants. This messaging system may occur over a network or all on a single machine.

DESQview/X

- Is a third DESQview (DESQview, DESQview 386 and now DESQview/X) adding distributed graphics and remote computing capabilities to Quarterdeck's multitasking DOS environment.
- Brings the X Window System to DOS — thereby enabling DOS users to participate in the many graphical and multimedia advances available today only on workstation platforms.
- Allows DOS PCs to participate in industry standard multivendor, multi-operating system distributed processing (cross-platform computing).
- Gives users of Unix X Window workstations access to many off-the-shelf DOS software programs.
- Gives DOS users access to Unix X Window Clients.
- Gives DOS users access to more powerful DOS programs that are not capable of being run on their own machine (remote DOS computing).
- Provides developers with an industry standard "open" graphical development environment and choice of user interface.
- Provides X Client developers and manufacturers access to the huge installed base of DOS machines.

Specifically, it

- Adds a graphical 3D look and feel to DESQview
- Gives users a choice of interfaces (window managers) the DESQview OSF/Motif or OPEN LOOK interfaces are prime examples
- Allows users to run many off the-shelf DOS applications in graphic w ndows side-by side with local or remote X Window Clients
- Runs either stand alone or as a networked system
- Allows local users to display and use many off the-shelf DOS applications that are run ning on another machine on a network
- Allows local users to display and use Unix graphical applications (X Clients) that are running on another machine on a network
- Allows local users to display and use DOS Clients written for X (graphical applications) that are running on another machine on a network.
- Allows X Window Unix users to display and use many off the-shelf DOS applications that are running on another DOS machine on a network
- Provides developers a platform to port X Clients from Unix to DOS.
- Allows developers to create X Clients in a DOS environment for later porting to other operating systems using the X Window System
- Gives developers a choice of appl cation styles OSF/Motif OPEN LOOK and others or the freedom to design their own

The purpose of this booklet is to explain how the features of DESQview/X are possible It provides coverage of the DESQview DOS multitasker, the X Window System and the integra tion of both of these technologies. This booklet also describes the development processes necessary to create X Client graphical applications for DOS

The intended audience for this document includes the following.

- End users and devlopers seeking an overview of the DESQview/X architecture
- Developers wishing to port X Clients to a DOS environment
- Developers wishing to create X Clients in a DOS environment for later porting to other X systems (typically Unix)
- Developers wishing to create large applications in a DOS environment (up to 16MB in size for a 286 processor machine 4 Gigabytes for a 386/486 processor)
- Developers wishing to create DOS applications using a standard graphical user interface (GUI)
- System Analysts interested in integrating DOS machines and other X Window machines on the same network
- System Analysts interested in gaining access to X Clients available on other architectures from a DOS based machine
- System Analysts interested in gaining access to DOS applications from X Terminals X Servers and any machine (includ ng other DOS systems) on a network
- System Analysts interested in integrating different types of machines with different architectures from different vendors on the same network each machine being able to utilize programs on other machines

Note that planned availability of products is first quarter 1991

2 DESQview and PC Fundamentals

This section describes the fundamentals of the DESQview multitasking software and basic PC concepts. Readers familiar with these topics are still encouraged to read the information presented here

2.1 Introduction

DESQview is a program that extends DOS (either PC DOS or MS-DOS) into a fully pre-emptive multitasking system. Contrary to popular belief, DESQview can perform multitasking on *all* classes of processor (8088, 8086 (PC XT), 286 (PC-AT), 386 and 486). In addition, the technical advances of the later processors empower DESQview with greater capabilities.

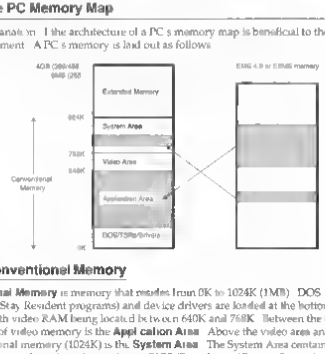
DESQview has an extremely high compatibility rate with current PC software - typically 95% better than the vast majority of other operating environments. DESQview can even run Microsoft Windows 3.0 and Windows programs. It can also run GEM-based programs.

Each program in the DESQview environment may run occupying the whole display screen or can appear in a small window framed by a border. Multiple applications may appear on the screen simultaneously each in its own window.

Certain applications may run in a small window and in background - depending on how the program has been written and the type of processor being used - a table later in this section summarizes specific capabilities.

2.2 The PC Memory Map

A brief explanation of the architecture of a PC's memory map is beneficial to the understanding of this document. A PC's memory is laid out as follows:



2.2.1 Conventional Memory

Conventional Memory is memory that resides from 0K to 1024K (1MB). DOS, TSRs (Terminate and Stay Resident programs) and device drivers are loaded at the bottom of this memory with video RAM being located between 640K and 768K. Between the top of DOS and the bottom of video memory is the **Application Area**. Above the video area and below the top of conventional memory (1024K) is the **System Area**. The System Area contains, for example special system code such as the machine's BIOS (Basic Input/Output System) or RAM for a hardware card. Typically this area is not contiguous, but contains portions of unused address space sometimes more than 128K in size.

2.2.2 Extended Memory

Extended Memory is memory that resides from 1MB and upwards (up to 16MB on a 286 or 4 Gigabytes on a 386/486). It is available on machines that use a 286 processor or better and hence is not available on 8088/8086 machines. To be able to directly access this memory the processor must be in a special mode called **protected mode**. This mode is incompatible with DOS and DOS applications which run in **real mode**.

2.2.3 Expanded Memory

Expanded Memory is memory that acts as a pool of memory which under the control of a special program (the **Expanded Memory Manager**) can be mapped into one or more conventional memory areas. **Mapping** is a process whereby a portion of expanded memory appears at a specific memory location through the use of special hardware (or in certain circumstances via software control). The system called Lotus Intel Microsoft Emulation of EMS Memory). Note that no transfer of data is actually performed. EMS memory is a bank-switch type of system (and hence very fast).

Unlike extended memory, expanded memory is available for all processor types. There are two different types of expanded memory: EMS 3.2 (LEMS and EMS 4.0).

2.3 EMS 3.2

EMS 3.2 currently map from 16K pages of memory (64K) into conventional memory at a time typically into a 64K area within the System Area called the **EMS Page Frame**. EMS 3.2 memory is essentially limited to enhancing the data handling capabilities of a program and has been superseded by the other two types of expanded memory.

2.4 EEMS and EMS 4.0

EEMS and **EMS 4.0** memory can map multiple pages of varying size at one time enhancing both data access and program execution capabilities a far more flexible scheme than EMS 3.2. Note however that some EMS 3.2 memory boards were packaged with a 4.0 Expanded Memory Manager. Unfortunately this gives the user an impression that EMS 4.0 memory is available with these boards when only 3.2 capabilities are.

Note that for the remainder of this document, all references to EMS 4.0 memory are also applicable to EEMS memory.

2.5 Software Emulation of EMS memory

Due to the technical capabilities of the 286 and 486 processors an Expanded Memory Manager like Quarterdeck's QEMM 386 can convert extended memory into EMS 4.0 memory. This is a better solution than using EMS 4.0 memory boards - expanded memory is much more expensive than extended memory due to the cost of the extra hardware needed to support the mapping capability. In the case of programs like QEMM 386 it's the Expanded Memory Manager that performs the mapping through software control.

2.6 Multitasking more than 640K

Despite DOS normally being limited to 640K for its programs DESQview can run more than 640K concurrently by using expanded memory. Programs are loaded first into the Application Area and when this is exhausted DESQview will load programs into EMS 4.0 memory. As DESQview task switches from one application to another it first maps the application from EMS memory into the Application Area and then runs it.

Note that EMS 3.2 memory is not used in this way due to the limitations of the specification. EMS 3.2 memory can only map a maximum of 64K and hence the available partition size is too small to contain the majority of programs.

On 80386 and 386 based systems it is essential to disable the hardware caching of video memory as low as possible, typically 256K, when using EMS 4.0 memory and DESQview. Some hardware has problems with these processors. MS 4.0 memory cannot be mapped into extended memory (LIM 4.0 only). Check your hardware manual. If in doubt, avoid memory intensive disabled DESQview commands and task applications when EMS 4.0 memory. In the case of VCPI 1.0 memory can act like EMS 3.2 memory however adapted programs (see the Program Swapping section for details) but may still have the LOADHI capability.

2.7 LOADHI capability

Since DESQview uses the Application Area below 640K to permit the multiple loading of programs it follows that the larger this area is the larger the applications can be that run in it. Therefore (unless an application is a DOS Extender application - see the DOS Extender section for details)

Any TSRs (Terminate and Stay Resident programs) or drivers (such as mouse or network drivers) that are loaded before DESQview occupy space in conventional memory. Once loaded DOS and reduce the amount of memory available to the Application Area. It is therefore advisable to keep the number of TSRs and device drivers using conventional memory to a minimum to ensure a maximum amount of space for applications.

One solution is available with the Quarterdeck QRAM, QEMM 50/60 and QEMM 386 products - the **LOADHI** capability. With this utility TSRs and device drivers can be loaded into the unused regions of the System Area thus freeing up more space below 640K and enabling larger applications to be run inside of DESQview.

2.8 Program Swapping

When all of the Applications Area and EMS 4.0 memory has been used to store programs any further loading of applications will cause DESQview to **swap** applications already running onto either a hard disk, a network drive, a RAM drive or expanded memory (even EMS 3.2 can support this type of operation). Any programs swapped out in this way will be suspended from running. Despite being suspended, any swapped applications can be swapped back in at the request of the user with only a short delay. Doing this will force one of the currently running programs to be swapped out in order to make room if there is not enough memory for the incoming program.

2.9 Switching and Windowing

Because DESQview can multitask multiple DOS applications a user can **switch** from one application to another using two or three keystrokes or mouse clicks. Because of this switching capability there is a concept in DESQview of one foreground application and multiple background applications.

Some applications may be running in **windows** smaller than the screen size and others may occupy the whole screen. In addition, some applications may be running in background while others are suspended. These capabilities are dependent on the video behavior of the program and the machine's processor type.

2.10 Application Video Behavior

DOS applications may be written to produce display output in either of two ways. They may call DOS and BIOS routines to perform the output or alternatively may write directly into the video area. The latter method is usually employed for speed reasons.

Applications that use the DOS and BIOS routines are termed **well-behaved** and others that write directly to the video area are **ill-behaved**. Typically graphical applications are ill-behaved; text based applications may be either.

2.10.1 Well-Behaved Applications

Since DESQview can easily intercept DOS and BIOS calls well-behaved applications may run in a small window or in background on any machine regardless of processor type When a well-behaved application makes a video BIOS or DOS call DESQview intercepts and executes the call but places the relevant information in a special save area called the **logical window buffer** as well as clipping and shifting the information to appear within a small window on the screen

2.10.2 Ill-Behaved Applications

Since ill-behaved applications write directly to the video RAM DESQview cannot run them in a small window or in background unless the processor is a 386 or a 486 (see the Virtualization section for details) These applications when run on an 8088/86 or 286 PC can only run full screen in the foreground Application developers should note that in many cases it is very easy to make an ill behaved text application well-behaved simply by adding a single subroutine call see "The SHADOW Call" section for details

2.10.3 Virtualization

Due to the sophisticated memory handling capabilities of the 386 and 486 processors DESQview can redirect an application that writes directly to video RAM to a portion of memory that DESQview calls the **logical window buffer** DESQview copies the information from this buffer to the actual video RAM clipping and shifting it as necessary to appear within a small window In this way DESQview can place with QEMM 386 and a 386 or 486 processor can **virtualize** ill behaved applications (including graphics ones) in small windows and run them in background

The only exception to this process are graphical DOS Extended applications see the DOS Extenders section for more information. This is because the virtualization process uses a special processor mode that is incompatible with the DOS Extender

2.10.4 Loaders

For machines that do not have a minimum of a 386 processor a loader may be available to run an ill-behaved text program in a small window and in background **Loaders** are utilities that alter a program's operation while it is running and coerce it into being well-behaved Quarterdeck supplies several loaders with DESQview for use with programs such as Lotus 1 2 3

2.11 Application Types

There are three types of application which exist in the DESQview multitasking environment DESQview-oblivious, DESQview-aware and DESQview-specific

DESQview-oblivious DESQview-oblivious programs know nothing about DESQview they include programs like Lotus 1 2 3 Microsoft Word or AutoCAD

DESQview-aware DESQview-aware program have been modified slightly to make them run more efficiently in DESQview Paradox dBASE and WordPerfect are examples of DESQview-aware programs

DESQview specific A DESQview-specific program has been written to take advantage of the DESQview API (Application Program Interface) Consequently these programs can run only when DESQview is present

2.12 DESQview API

Every copy of DESQview includes the **DESQview API** (Application Program Interface). This interface allows programs to call the DESQview subroutines and functions in order to start and close down other applications, move, resize and scroll their windows, perform interask communication and many other functions.

This interface is callable by Assembler, C, BASIC, Pascal, Clipper and dBASE programs.

2.12.1 The SHADOW Call

One API call of particular significance is the **SHADOW** call. This call may be made when DESQview is either present or not.

An ill-behaved text application will typically determine the kind of system present (monochrome or color) and load a variable with the corresponding video RAM value for this system (either B000H or B800H). From then on, the application will use the variable in order to access video RAM.

If during a program's initialization a register is loaded with the desired video RAM value and the SHADOW call is performed before storing it, then this will enable the program to become well behaved under DESQview. When using the SHADOW call, DESQview returns the logical window buffer for that application, whereas under DOS the value remains unchanged.

Since the application stores the returned value and uses it whenever video RAM access is required, the application is writing directly into the DESQview logical window buffer instead. It is to the screen. DESQview **shadows** from the logical window buffer to the screen, clipping and shifting as necessary so that the otherwise ill-behaved text application can run in a small window and in background on all processor types. This process is fast enough to be rarely noticeable by the user. Word Perfect, Dbase, and Paradox are examples of commercially available programs that do this.

2.13 Processor Types and Modes

Since the original PC was introduced in 1981, various processors have been used, each improving upon the previous version and providing greater functionality. This functionality was always gained with the advantage of backward compatibility with all the previous processors.

2.13.1 The 8088/8086 and Real Mode

When the PC was first introduced, it used an Intel 8088 microprocessor. This was a 16-bit architecture processor with a segmented memory scheme capable of addressing 1MB. The 8088 used an 8 bit external data path unlike its otherwise functionally equivalent bigger brother, the 8086 which used an external data path of 16 bits. The mode of operation of these two processors is termed **real mode**.

2.13.2 The 286 and Protected Mode

The 286 processor supplies a real mode capability but improved upon the 8088/86 by providing a new mode called **protected mode**. In protected mode the 286 can access up to 16MB of memory (again by using a segmented addressing scheme) however certain operations available in real mode (such as segment arithmetic) are prohibited in protected mode. In addition, protected mode also has the hardware necessary for an operating system to protect an application from crashing the system or overwriting another application.

Unfortunately protected mode is sufficiently different from real mode that DOS and regular DOS applications cannot operate in protected mode. For a long time this limited applications running in real mode and hence constrained them to the 1MB limit. Thankfully a solution has become available that addresses this called the DOS Extender - see the 'DOS Extenders' section for details.

2.13.3 The 386 and V8086 Mode

After the 286, the 386 processor was introduced next. In providing backward compatibility the 386 has both a real mode and protected mode capability. In addition to this Intel added a third mode called **Virtual 8086 Mode** that can operate under the auspices of protected mode. This mode supplies a virtual 1MB 8086 style environment while running in protected mode. This elegant solution enables regular DOS and real mode applications to run under protected mode without modification. The 386 also has an effective addressing range of 4 Gigabytes. It supports a flat memory model as well as a segmented addressing mechanism.

Also included in the 386 were memory mapping capabilities, a 32-bit architecture and hooks for a paged virtual memory system as opposed to a somewhat meager segmented virtual memory system that became available in the 286.

Note that 386 technology has been realized in two processors: the 386SX and 386DX. The 386SX processor uses the 286 32-bit architecture internally, yet has a 16-bit external data path. The 386DX processor (basically a renamed version of the original 386 and often referred to as the 386 processor) uses an internal 32-bit architecture and a 32 bit external data path.

These two processors are functionally equivalent and are referred to by the 386 moniker for the rest of this document.

2.13.4 The 486

The latest member of the 80x86 family to be introduced has been the 486 processor. Basically this is similar to a 386 processor with the inclusion of a math coprocessor (an optional external component with all the other previous processors) and an internal memory cache. It is also faster due to improved instruction execution.

2.14 DOS Extenders

Since DOS cannot run in protected mode, a way was devised for protected mode applications to run under DOS and use other real mode services. Protected mode applications are desirable since they have access up to 16MB of memory on a 286 and 4 Gigabytes on a 386/486.

A **DOS Extender** is a special utility that is linked in to a protected mode application. Whenever the application makes a DOS call or any other request that requires real mode, the DOS Extender copies down any necessary data into the 1MB conventional memory area and switches into real mode. It then calls the requested function and on return switches back into protected mode, returning any results to the protected mode application.

Note that there are two types of DOS Extenders: 286 DOS Extenders and 386 DOS Extenders.

- 286 DOS Extenders run 16-bit protected mode applications on a machine with a minimum of a 286 processor. They have access to a 16MB address space.
- 386 DOS Extenders run 32-bit protected mode applications on a machine with a minimum of a 386 processor. They have access to a 4GB address space.

Most DOS Extenders also have a **virtual memory** option. That is, a DOS Extended application may run in less memory than is normally available by using virtual memory techniques.

In essence, the DOS Extender becomes the system's **control program**. This would normally have posed a problem to DESQview as protected mode allows only one control program in a system. Since DESQview will multi tasks applications, multiple DOS Extended applications would conflict with each other because each expects to be the control program. This is compounded by the fact that DESQview 386 (DESQview plus QEMM 386) is also a control program.

To obviate this problem, Quarterdeck and Phar Lap (one of the companies that produce DOS Extenders) developed the VCPI (Virtual Control Program Interface) specification which has been adopted by all major 386 DOS Extender manufacturers. See the "VCPI Specification" section for details.

It should be noted that VCPI is a specification for 386 and 486 processors yet 16-bit protected mode applications may be run on 286 machines. In order for DESQview to multitask multiple 16-bit protected mode programs on a 286 so that they do not assume control of the same blocks of extended memory their individual 286 DOS Extenders must use a new memory specification **XMS** (Extended Memory Specification). Most DOS Extenders either have been updated to use XMS services or are in the process of being updated. A utility program called QEXT.SYS supplies the necessary XMS services for 286 DOS Extenders running under DESQview on a 286 machine, whereas QEMM 386 supplies the services for DESQview 386.

Note that a DOS Extended application consists of two parts. A real mode portion of the DOS Extender resides in conventional memory and interfaces with the protected mode portion that resides with the protected mode application in extended memory. When DESQview performs a task switch to a DOS Extended application, control is passed to the real mode portion of the application is mapped into the conventional memory Application Area and that the protected mode portion is visible in extended memory. Since the majority of the application resides in extended memory and only a small portion (the real mode part) need occupy the Application Area, DOS Extended applications tend not to be constrained by the size of the Application Area as regular real mode applications are.

2.15 VCPI Specification

The **VCPI** specification was developed so that multiple protected mode control programs can coexist and interact within a single 386 (or 486) system. The specification consists of two parts: a VCPI server and several VCPI clients. The VCPI clients request memory and mode switching services from the VCPI server.

In a DESQview 386 system, the VCPI server is implemented within QEMM 386 and the DOS Extended applications become VCPI clients. Whenever a DOS Extended application requires memory services it calls upon the VCPI server to perform them. When QEMM 386 is not present, the DOS Extender performs all services for itself. The end result is that DESQview is able to run a mix of real mode and DOS Extended (protected mode) applications concurrently on a 386/486.

As mentioned earlier, 286 machines may run multiple 286 DOS Extended applications only if the DOS Extenders utilize XMS services.

2.16 DESQview Capabilities

DESQview's ability to window an application and run it in the background is a function of the machine's processor and the type of application. Here are the possible combinations:

DESQview Capabilities		8088/8086	286	386/486
Well-behaved Text Application or Ill-behaved Text Application with Loader	Real Mode	W	W	W
	16 bit Protected		W	W
	32 bit Protected			W
Ill-behaved Text Application	Real Mode	F	F	W
	16 bit Protected		F	WF
	32 bit Protected			WF
Graphics Application	Real Mode	F	F	W
	16 bit Protected		F	F
	32 bit Protected			F

W — Application can run in a window and in background
F — Application can only run full screen and not in background
WF — Dependent on individual application; must run as W, some may run as F
Processor cannot support this type of application

Note that the majority of DOS applications operate in real mode.

3 An Introduction to X

This basic concepts of the X Window System are described in this section. Readers familiar with X may elect to skip this section.

The X Window System is a powerful concept that utilizes machine and device independence as well as providing a graphical interface to users with both keyboard and mouse support.

3.1 Traditional Graphics Output

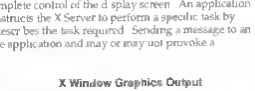
In traditional systems, if an application wishes to produce graphical output on a computer's display device, it will typically call a library system software graphic subroutine. This subroutine performs the task requested (in the example shown, draw a line) and once the task has completed, control returns to the application.

3.2 X Servers, Clients and Protocol

In an X Windows system, the system software is replaced by an application called the **X Server**. It is this application that has complete control of the display screen. An application that wishes to produce graphical output instructs the X Server to perform a specific task by sending it an information message that describes the task required. Sending a message to an X Server returns control immediately to the application and may or may not provoke a response from the Server.

The different types of messages are collectively called the **X Protocol**. One message draws a line, another a circle and yet another may print some text.

Any application that displays graphical output by sending X Protocol messages is labelled an **X Client** in contrast to an application that uses some other means.



In return the X Server may send back to an X Client special messages such as event messages or error messages. These special messages are also part of the X Protocol.

X Clients typically create windows for their output. It is quite feasible (and generally the case) that a single X Client will create and utilize several windows on the X Server's screen simultaneously.

Note that an X Server may handle the graphics output for multiple X Clients concurrently and only understands X Protocol requests as a means to produce graphical output. It usually does not produce graphics output any other way.

3.3 An Event-Driven System

X is an **event-driven** system. That is, X Clients are typically suspended until an action occurs on the X Server for which they have a vested interest. X Clients are restarted by the X Server sending them special X Protocol messages. These event messages include ones that instruct an X Client to redraw its window (for example if a part of its window becomes uncovered by the movement of another window), that a window's size has changed or that a key has been pressed. An X Client processes these messages producing whatever output may be necessary and then returns to a suspended state until another message is received.

This is in direct contrast to the way traditional applications have been written. Those applications are **procedure-oriented** and are written to assume an active role in the interrelationship between the user and the program. Typically the program will steer the user through the execution of the task at hand. For example, the user down a narrow set of predefined procedures. The program only accepts input (be it keystrokes or mouse clicks) from the user at predictable times. An order entry application is a good example of a procedure-oriented program.

Event-driven applications take a more passive role in that they respond to input from the user or the system at unpredictable times. This type of application can provide a more flexible framework within which a user may operate. Typically there are no predefined procedures and many ways to complete a task. A user is free to use whatever tools the application provides and in any manner desired to achieve the final result. A drawing/designing program is a good example of an event-driven application.

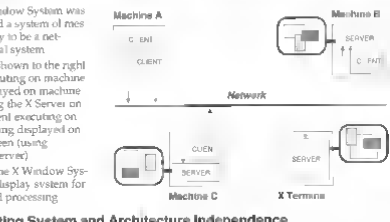
3.4 A Distributed System

Since the X Client communicates with the X Server through information messages, it is possible for the X Protocol requests to be sent over a network to an X Server running on a different machine.

In fact the X Window System was designed around a system of messages specifically to be a fast networked graphical system.

In the diagram shown to the right an X Client executing on machine A is being displayed on machine C's screen (using the X Server on C) and an X Client executing on machine C is being displayed on machine B's screen (using machine B's X Server).

Consequently, the X Window System is an ideal display system for truly distributed processing.



3.5 Operating System and Architecture Independence

None of the machines need be from the same vendor running the same operating system since all communication between X Clients and Servers is performed over a network using a well-defined message protocol (the X Protocol). Naturally a program cannot be copied to a different type of machine on the network and subsequently run. It would have to be recompiled for a different machine's architecture/operating system. Despite this there is now a fully integrated distributed open system. And graphical too.

3.6 X Terminals

In the previous diagram one of the clients running on machine A is being displayed by a special machine that only has an X Server running on it. In effect a thin, as a remote graphics terminal to machine A. This type of machine is called an **X Terminal** and is sole purpose is to display graphics from X Clients running on remote machines.

Typically the majority of PC implementations of the X Window System have been as X Terminals. PCs are notorious for memory limitations and hence an X Server application would normally occupy all of the PC's memory. With the advent of DESQview/X, however, PCs can run X Servers, DOS applications and X Clients simultaneously.

3.7 A Stand-Alone System

In the previous diagram, machine B's X Server was displaying output from an X Client running on machine C, and it also displaying graphics from an X Client running on itself. In this case, the X Protocol messages are not sent out over the network to another X Server but are routed within the machine to the local X Server.

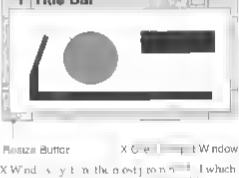
This concept can be extended to include a scenario where the machine is not connected to a network - all X Clients run locally and are displayed by the local X Server. Naturally this requires a multitasking operating system such as DOS with DESQview.

3.8 The Window Manager

The X Server only produces graphical output according to X Protocol requests and does not provide functions for the user to control the size, position and stacking order of the displayed windows.

These functions could have been implemented within each X Client but would have led to much redundant programming. They could have been implemented within the X Server itself but the designers of the X Window System took a more flexible approach.

A special X Client is run (either locally or remotely) for each X Server called the **window manager**. This program is given special privileges and is allowed to superimpose all of the windows being displayed by the X Server. The window manager will typically place some form of window decoration around the outside of each X Client window so that includes resize and move buttons as well as a title bar. It then becomes a function of the window manager to resize, move, rearrange a window according to the wishes of the user by mouse clicks on a window's decoration or selections from a window manager menu.



At present there are several managers for the X Window System, each of which are the OSF/Motif OPEN LOOK and the Tab (previously known as Tom's) Window Managers. DESQview/X also supplies its own window manager, DWM.

Due to the design of the X Window System, a window manager may be closed down and another may be subsequently started without affecting any of the X Clients being displayed on the screen[1]. The old window decorations disappear from the screen and are replaced by new decorations created by the incoming window manager.

[1] Note that the window manager only creates the 'look and feel' of an X Client with regards to its window decoration. Whatever an X Client chooses to display in its output window is independent of the window manager. Program libraries are available to X Client developers that allow them to create an application with a specific look - either an OPEN LOOK or OSF/Motif look, for example. These program libraries are called toolkits and are explored in the next section.

3.9 X Development Layers

In order to create an X Client, a developer will call upon a variety of program libraries for assistance. For DESQview/X these libraries are presently linked into each X Client, though they may become a shared resource among all X Clients on a system.

3.1 Xlib

For an X Client to be able to communicate with an X Server it needs to generate X Protocol requests for transmission to the Server. Building these requests can be cumbersome and hence a library was created called **Xlib**. Xlib is (generally) the lowest level interface that an X Client uses to communicate with the X Server. It is a set of C subroutines that, for the most part, are a one-to-one mapping from C to X Protocol requests though some Xlib functions can generate multiple X Protocol requests.

For example if an X Client uses the function XDrawLine it calls the appropriate corresponding Xlib which builds a PolySegment request and transmits it to the X Server.

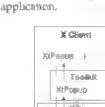
Note that the Xlib library imparts no specific 'look and feel' to an X Client - it merely consists of requests to create a window, draw a line, print some text, etc. The appearance of an application is generally determined by another program library - a toolkit.

X Clients may be written so that they use only Xlib and no other program libraries (toolkits).

3.2 Toolkits

Since Xlib is rather low-level in the scope of its operations, another program layer may exist on top of Xlib - the **toolkit**. Toolkits generally have routines for building menus, push buttons, slider controls and the like. Since the toolkit generates these basic window components for the X Client, it is the toolkit which creates the actual 'look' of an application.

An individual toolkit function may call several Xlib functions which in turn can create multiple X Protocol requests.



For example if the X Client wishes to make a popup window appear it could call (using one specific toolkit) XtPopup to perform the function. XtPopup in turn makes several Xlib calls which may generate multiple X Protocol requests.

An X Client may still (and often does) call Xlib functions even if it uses a toolkit.

Some of the more prominent toolkits that are generally available are:

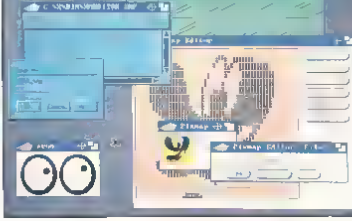
Athena Toolkit	A fairly rudimentary toolkit supplied by MIT.
OSF/Motif Toolkit	This toolkit is supplied by the Open Software Foundation and provides a sculptured 3D look. This toolkit (and its complementary window manager) is promoted by a consortium of companies (the OSF) that include DEC, Hewlett Packard and Microsoft.
Xol Toolkit	A toolkit conforming to the OPEN LOOK* graphical user interface standard - seen to become 3D in appearance. This toolkit and its OPEN LOOK window manager is supplied and promoted by AT&T.
Xview Toolkit	A toolkit conforming to the OPEN LOOK* graphical user interface standard - but with a different programming interface (SunView) than Xol. This toolkit is supplied and promoted by Sun Microsystems.

*Note that OPEN LOOK is not a toolkit or window manager in itself - it is merely a design specification for the appearance of a user interface. Xol and Xview are toolkits that adhere to this specification and hence create the same look and feel.

Xt Intrinsics Toolkit is of particular interest in that applications written using its routines may be ported to other non X Window systems such as Apple's Macintosh Windows or OS/2. Unlike Tcl/Tk, most toolkits which only call Xlib. It accomplishes display output by using libraries that map generic toolkit calls into the functions used by the desired target system (Xlib or otherwise).

XVT Toolkit Programs generated with this toolkit may use either the Xlib, OSF/Motif or OPEN LOOK libraries to produce applications with an OSF/Motif or OPEN LOOK appearance. In addition to this, XVT applications may be linked with Macintosh Windows or Presentation Manager (OS/2) libraries to produce applications that run under those display systems.

The following screen shots should help to illustrate the concepts of a Window Manager, Toolkits and the like.



The X Clients in the above picture are:

QOS Background An X client written using only Xlib. XVT libraries are used to produce the output window.

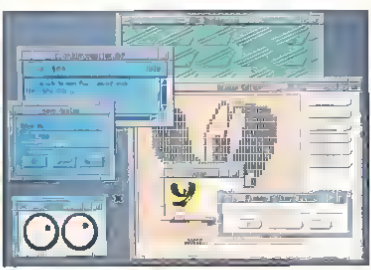
XMEDITOR EXP Text Editor This client was written using the OSF/Motif Xlib. The X Client is run in background mode between windows and editor and a save As will be.

Pixmap Editor OPEN LOOK has characteristic buttons with rounded ends. The XVT toolkit, however, does not have these rounded buttons. The X Client is currently displaying three windows: the edit window, the tutorial pixmap, the pixmap (actual size) and an Open File box.

Xeyes This X client uses the Athena Toolkit to display a pair of eyes that follow the mouse around the screen.

All of these X Clients are running under the control of the Tab window manager. The window manager has placed a frame around many of the windows on the screen, along with controls such as resize and iconify buttons.

If the Tab window manager is closed down and the OSF/Motif window manager is started, the following display appears:



D[illegible] t[illegible] change in w[illegible]nd[illegible]w manager, the X Cl[illegible]nts window contents remain the same. Only [illegible] window decoration has changed in this last [illegible]an OSF/Motif 3D effect.

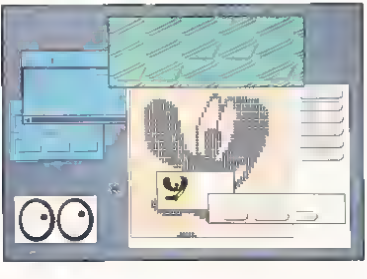
W[illegible] the OSF/Motif wlod[illegible]w manager acti[illegible]thi[illegible] X Client built using the OSF/Motif toolkit (XMHDR/OR EXP) blend[illegible]s well with env[illegible]nment since it has the same appearance style[illegible]as the window manager

If the OSF/Motif window manager is replaced by [illegible]s OPEN LOOK window manager the [illegible] look display appears

Once again a change in window managers does not change the contents of the X Clients window only the window decoration has altered

In contrast flash on to the OSF/Motif-style editor and the OSF/Motif window manager can be seen in the previous picture that the Pixmap Editor s windows complement the Xol window manager s window decoration This is because both products were built using the Xol Toolkit and hence have an OPEN LOOK appearance

Although it is unlikely that a user would want to run X Clients without a window manager the following picture shows how this would appear



As can be seen above, the usefulness of having no window manager is debatable but by no means impossible Having no window manager active would most probably occur when only one X Client is being run on an X Server

These pictures highlight the concept of a window manager as being a special X Client that decorates the outside of all other X Clients windows and allows a user to control their size, position and ordering on the screen. The pictures also show how a toolkit influences the look and feel of an X Client and how its appearance is independent of the active window manager

3 9 3 Intrinsics and Widgets

Some toolkits may tools be regarded as a single entity but others are conceptually split into two parts. One of these parts is termed the Intrinsics Library and the other part a Widget Library

The Widget Library

Widgets are abstract data objects such as buttons scrollbars and other such objects. An X Client can be easily constructed from a number of widgets The X Client does not have direct control of the actual appearance of a widget only its general form, size or contents The appearance is determined by the toolkit

A **Widget Library** uses both function calls in the Intrinsics Library as well as Xlib.

The Intrinsics Library

The **Intrinsics Library** provides an object-oriented framework on which a Widget Library depends. It handles the creation, deletion and management of widgets as well as their event message handling.

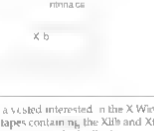
It is possible for an X Client to call the Intrinsics library directly as well as the Widget Library (and of course Xlib).

The Athena, OSF/Motif and Xol Toolkits consist of a Widget Library and an Intrinsic Library. In the case of OSF/Motif and Xol, they use a modified form of the Athena Toolkit's Intrinsics library, Xt.

3.9.4 Xt Intrinsics

In some cases an X Client may only call the Xt Intrinsics and Xlib libraries. This type of application dispenses with a widget library, supporting its own unique set of abstract data objects (widgets) that are manipulated and managed by Xt.

An application that does this is able to provide its own look and feel, all the while saving its developer time and effort by using the object or event functions of Xt.



3.10 X11

MIT, in association with a consortium of companies who have a vested interest in the X Window System (The X Consortium), releases MIT X distribution tapes containing the Xlib and Xt libraries as well as sample X Clients and an X Server. It is these tapes on which all other toolkits and X products are based.

The current release of these distribution tapes is the X11 release, revision 4 of the X Window System, otherwise known as X11 R4.

3.11 Toolkit Summary

Here follows a table of the Toolkits discussed in this document in summary form for quick reference:

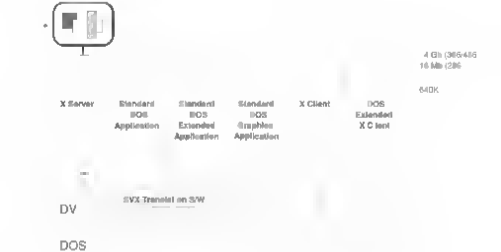
Toolkit	Supplier/Promoter	API	Interface
Athena	MIT	Xt	(rudimentary)
OSF/Motif	OSF	Modified Xt	OSF/Motif
Xol	AT&T	Modified Xt	OPEN LOOK
Xview	Sun	SunView	OPEN LOOK
XVT	XVT	XVT	(various)

4 DESQview/X

This section describes how the X Windows System is integrated into the DESQview environment and highlights the capabilities of the combined system.

4.1 General Structure

The general structure of a stand-alone DESQview/X system is shown below.



DESQview is loaded on top of DOS, the first program booted into the computer. Multitasking within DESQview can be several program partitions. In the case of DESQview/X, one contains an X Server.

4.2 The X Server

Display output for the system is provided by the X Server product. The X Server is run within a DESQview partition and is multitasked along with all of the other programs in the system.

The X Server controls the display screen (for the most part) and hence the display resolution of the system and compatible display types are determined by the X Server and not by DESQview. Currently EGA, VGA, Extended VGA, 8514 and DGIS displays are supported.

The X Server may be run as either a real mode application (up to 640K) or as DOS Extended application (up to 16MB). Typically the DOS Extended application version is run, thus giving the X Server more workspace to perform its display functions and enables it to handle more windows concurrently.

4.3 Standard DOS Applications

If a standard DOS application (for example Word Perfect or Lotus 1-2-3 release 2) is running within the system, its display output is translated dynamically (that is, on-the-fly) by special DESQview/X Translation Software into X Protocol requests and routed to the X Server for output.

In effect, a DOS application is made to appear like a regular X Client.

DESQview/X does this for well-behaved applications by trapping the BIOS and DOS calls they make and converting them into the equivalent X Protocol requests

In the case of ill-behaved applications DESQview/X **virtualizes** the application. DESQview/X remaps the application's video RAM to an unused portion of memory and scans the **logical window buffer** for any changes producing X Protocol requests from the scanning process

Note however that this process requires a minimum of a 386 processor the 286 and 8088/8086 processors lack the necessary hardware to perform the remapping operation

On a 286 or 8088/8086 machine DESQview/X will run any ill-behaved applications independently of the X Server each occupying their own full screen with the user being able to switch between the X Server screen and the ill-behaved DOS application screens

Note that an ill-behaved application that has a corresponding loader program becomes well behaved with respect to DESQview/X (for example Lotus 123)

4.4 Standard DOS Extended Applications

If a standard DOS Extended application (for example Lotus 1-2-3 release 3 or Paradox 386) is running within the system, it is treated much the same as a standard DOS applicati n. The major difference being that DOS Extended applications have a far greater workspace available to them than do regular DOS applications (up to 16MB on a 286 4GB on a 386/486)

4.5 Standard DOS Graphical Applications

Translating a DOS application's graphics screen into X Protocol requests is theoretically possible but would be too slow in operation to be realistically useful

Consequently DESQview/X bypasses the regular translation process for standard DOS graphical applications (like Ventura Publisher) and places the information inside the relevant window leaving displayed by the X Server

Since graphical DOS applications are ill-behaved DOS applications they are subject to the same constraints they are only w ndowable on 386/486 systems otherwise they must run independently as full screen applications

DOS graphical applications that an. DOS Extended may run only as full screen applications even with the availability of a 386/486 processor In order to remap their video RAM (as is needed in the case of an ill behaved application) DESQview 386 requires the use of a special processor mode (virtual 8086) that is incompatible with the DOS Extender

4.6 X Clients

X Clients may be running on a DESQv ew/X machine in one of three forms If small enough to fit within the conventional memory Appl cation Area they may run in real mode If larger they require a DOS Extender to res de in the system If the X Client is a 16-bit protected mode application, it requires a m n mum of a 286 processor and may be as large as 16MB If it is a 32 bit protected mode application the X Client may (theoretically) be as large as 4GB on a 386/486 system

Since X Clients already produce X Protocol requests (unlike DOS applications) they need no translation software Instead, their X Protocol requests are routed directly to the X Server for display output

These X Clients may be ported from other X platforms (such as many Unix machines) or else may be developed d rectly under DESQview/X How an X Client is either ported or developed to function under DESQview/X is described in the section "Development Issues

It is important to remember that an X Client is similar to a DOS graphical application in that it produces graphical output but is very different in the way it achieves this DOS graphical applications usually write directly to the video RAM an X Client uses X Protocol requests to an X Server to produce the same effect As a result, windowing of an X Client is always possible

4.7 The Window Manager

Since a user will want to control the windows displayed on the screen by the X Server a minimum form of X Client will normally be run in a DESQview/X system: the window manager. The window manager available, includes:

DWM — The DESQview Window Manager provides a 3D sculptured look and has a menu system very similar to the regular DESQview menu system. One of its most important characteristics is that it is under 50K in size.

OSF/Motif — Using a 3D sculptured look, this manager is a DOS Extended X Client, approximately 650K in size.

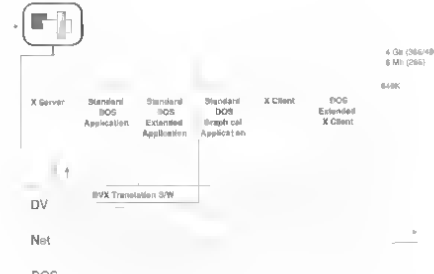
Tab (formerly Tom) — A fairly simple window manager (appearance-wise) approximately 300K in size.

Xol is OPEN LOOK — This window manager implements the OPEN LOOK graphical user interface. It currently consists of 3 separate X Clients: a window manager, a workspace manager and a file manager. In all, it is 1.5MB in size.

Xview is OPEN LOOK — A window manager that also implements the OPEN LOOK graphical user interface. It has been written using the Xview Toolkit.

4.8 The Network Connection

When a DESQview/X system is connected to a network the structure is identical to that of a stand-alone system, but with the inclusion of network software:



Currently DESQview/X can communicate over networks using the TCP/IP protocol. Others may be available soon.

Would run a Client application (a DOS application translated to X Protocol requests) is started under DESQview/X (e.g. by typing the name supplied that specifies which screen the program's output should be displayed on). The standard procedure for any X Window System) If the display specified is the local DESQview/X screen DESQview/X will route the X Protocol requests to the correct machine on the network via the network software If on the other hand the user requested the display on the local screen DESQview/X will route the X Protocol requests directly to the local X Server in the case of the stand-alone system.

DESQview/X will then route the network sends X Protocol requests to the DESQview/X system for display on that screen DESQview will accept the messages and route them to its X Server.

In the previous diagram the X Client and the Standard DOS Application (after translation) are sending X Protocol requests out over the network while the other three applications are being displayed on the local display. X Protocol requests are also being received from remote X Clients over the network and are being routed to the local X Server.

When connected to a network DESQview/X provides some additional features the remote shell feature and remote clients.

4.8.1 Remote Shell

A **remote shell** is one method for starting up applications on remote (other) machines anywhere on the network. This may at first appear to have limited use why for example would a DESQview/X user want to start up an application on another machine be it Unix or DESQview/X Or conversely why would a Unix user want to start up an application on a DESQview/X (DOS) machine?

The fact that X Protocol requests produced by an X Client may be routed to any X Server on the network provides the answer to these questions A user seated at one machine (be it DESQview/X or Unix) may use the remote shell feature to start up an application on another machine yet have its output appear on the user's local machine (or any other display on the network) The user is now able to operate and use the X Client that is running remotely X Clients that are run this way are termed remote clients

4.8.2 Remote Clients

Remote clients open up a wealth of possibilities for users connected over a network All of the X Clients on an X network can be started up and used by any X Window user on the system

Since regular DOS application screens are dynamically converted to X Protocol requests by DESQview/X the DOS applications appear on a network as X Clients Because of this non-DOS users on a network may use the DOS applications available on a DESQview/X machine DOS applications that may not be used this way are those which cannot be translated into X Protocol requests on the host DESQview/X system That is, ill behaved applications when the host processor is only an 8088/86 or 286 and standard DOS graphical applications

In effect any DESQview/X machines on a network appear somewhat as Unix machines running multiple X Clients.

The reverse is also applicable a networked DESQview/X machine may use X Clients available on other non DOS machines (for example a Sun or SCO Unix system)

Shown on the next page is an example of one instance of a multivendor network with DESQview/X machines.

Numbers in parentheses indicate the host machine on which an application is running. Numbers in brackets [] indicate the target machine where an application's output is received. All applications listed are a trademark or registered trademark of their respective manufacturers.

4.9 Stand-Alone or Networked?

DESQview/X may be run either as a stand-alone system or networked.

If run as a stand-alone system, applications typically run on the system will be the X Server (for display output) and a window manager (to control the windows); multiple DOS applications and multiple X Clients.

If a DESQview/X machine is networked, the maximum required memory is the X Server. The window manager and any X Clients (both regular X Clients or dynamically translated DOS applications) on that or any other DESQview/X machine may all be run remotely on other machines on the network. Usually, some local applications will also be run.

4.9.1 Unix Machines and DOS programs

Assume a network to primarily consist of Unix machines and/or X terminals. If a DOS machine with a powerful processor (such as a 386 or 486) is added to the network, all of the Unix X Windows users on the network would then be able to use many of the DOS applications that are available on the DOS machine. For example, running DESQview/X:

4.9.2 DOS Machines and Unix Programs

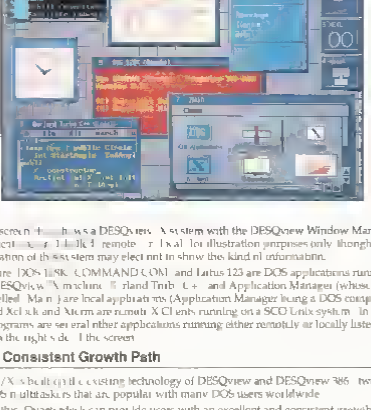
The converse of the above situation is also true. On a DOS-based (DESQview/X) network, the addition of a Unix machine provides the DOS users access to any X Clients on the Unix machine. Large, powerful applications now become feasible that previously were not available to DOS users and which would have suffered running under a slower processor.

4.9.3 16-bit DOS Machines and 32-bit DOS Programs

Another scenario of interest involves a network of 286 (or even possibly 8088/8086) DOS machines. Adding a powerful 386 or 486 DESQview/X system to the network would enable all users to run 32-bit protected mode programs - applications that normally are not capable of being run on this class of machines.

4.10 A User's View

It is not important to the user whether an application being used is running locally or remotely. It is possible with DESQview/X to hide all of these details so that a user sees a screen much like the following:



The above screen shows a DESQview/X session with the DESQview Window Manager. Some applications are running locally and remote or local (for illustration purposes only, though a real implementation of this system may elect not to show this kind of information).

In the picture DOS tasks, COMMAND.COM and Lotus 123 are DOS applications running on a remote DESQview/X machine. Borland C++ and Application Manager (whose window is labelled "Manager") are local applications. (Application Manager being a DOS compiled X Client) and Xclock and Xterm are normal X Clients running on a SCO Unix system. In addition to these programs are several other applications running either remotely or locally listed as consoles on the right side of the screen.

4.11 A Consistent Growth Path

DESQview/X is built on the existing technology of DESQview and DESQview/386 - two time-proven DOS multitasking systems that are popular with many DOS users worldwide.

Because of this, Quarterdeck can provide users with an excellent and consistent growth path that starts with DESQview.

4.11.1 DESQview

DESQview provides the DOS user with a multitasking environment on machines with as little as 640K, a hard disk, a monochrome monitor and an 8088 processor. DESQview is a character based environment but can also run graphics applications. Features include windowing and program swapping as well a keyboard macro program, a help system, a DOS Services utility and easy to use keyboard or mouse control.

4.11.2 DESQview 386

In addition to DESQview's features, DESQview 386 provides the 386/486 DOS user with a multitasking environment that incorporates excellent memory handling, coupled with superior windowing features and program protection.

4.11.3 DESQview/X and DWM

DESQview/X and the DESQview Window Manager - the graphical DESQview - DESQview/X incorporates all of the functionality of DESQview and DESQview 386, yet sports a graphical interface that is consistent with the menuing system used by these products. Because of its complete X Window capability, DESQview/X also gives users the capability to run local X Clients as well as access to DOS applications and X Clients on remote machines (using the appropriate network software).

4.11.4 DESQview/X and OSF/Motif,OPEN LOOK Window Managers

When DESQview/X is joined by either the OSF/Motif or an OPEN LOOK window manager, users will have a consistent look and feel across all machines on a network - from DOS machines, DESQview/X to X-term-like Unix workstations, minis and mainframes.

4.12 DESQview/X System Capabilities

The capabilities of a DESQview/X system outlined in the preceding sections is dependent on the machine's processor. Consequently, here is a table of the DESQview/X capabilities versus processor type:

DESQview/X Capabilities		286	386	386/486
Remote X Client or DESQview/X DOS Application		W	W	W
Local X Client	Real Mode	W	W	W
	16-bit Protected	—	W	W
	32-bit Protected	—	—	W
Local Well-behaved DOS Text Application or Local Ill-behaved DOS Text Application with Loader	Real Mode	W	W	W
	16-bit Protected	—	W	W
	32-bit Protected	—	—	W
Local Ill-behaved DOS Text Application	Real Mode	F	F	W
	16-bit Protected		F	WF
	32-bit Protected	—	—	WF
Local DOS Graphical Application	Real Mode	F	F	W*
	16-bit Protected		F	F
	32-bit Protected			F

W — Application can be displayed in an X window and can act as a remote X Client for another X Server on a network.

F — Application can only be displayed full screen and cannot act as a remote X Client for another X Server on a network.

WF — Dependent on individual application - most run as W, some may run as F.

W* — Application can be displayed in a local X window only.

— Processor cannot support this type of application.

4 13 System Configuration

A minimum of a 286 processor and 2MB of memory is recommended to run DESQview/X

Note however that it is possible to configure a DESQview/X system with less than these requirements

For a 286 system the mix of extended memory versus expanded memory depends on the applications that will be run locally on the system As a rough guide real mode applications use expanded memory while 16-bit protected mode (DOS Extended) applications use extended memory

For 386/486 systems the memory calculation is vastly simplified only extended memory is required A quick calculation would be

Total memory in system (including motherboard)	=	1MB (for system software)	+	Total amount of memory needed to run required applications concurrently

The DESQview/X system software consists of DESQview QEMM 386 the X Server product the DWM window manager and other supporting software

4 14 Summary of DESQview/X

The preceding sections show that DESQview/X has many worthwhile features which are summarized below

DESQview/X

- Adds a graphical 3D look and feel to DESQview
- Gives users a choice of window managers (look and feels)
- Provides a growth path from character mode DOS to industry standard graphical user interfaces
- Runs most regular DOS applications
- Runs DOS Extended applications up to 16MB (286) or even 4GB (386/486)
- Can display DOS applications within graphical windows
- Provides access to the wealth of DOS applications only other machines (DOS or non DOS) on a network
- Multitasks DOS applications and X Clients either locally or remotely
- Provides DOS users with access to powerful X Window applications available on larger machines
- Provides X Client developers with a user base of millions of DOS machines
- Provides a simple port of applications from other X systems to DESQview/X and vice versa
- Gives developers a choice of application appearances based on toolkit chosen
- Supports toolkits that provide features required by the serious developer push buttons scroll bars pop up dialogue boxes and the like
- Supports a wide range of PC hardware

5 Development Issues

As discussed in an earlier section there are three different types of applications that may be generated for a DOS system: real mode 16-bit protected mode and 32-bit protected mode programs. The latter two types (the protected mode programs) require a DOS Extender in order to run within a DOS system.

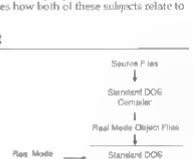
This section outlines the development procedures for the different program types takes a look at the more popular DOS Extenders available and examines how both of these subjects relate to the development of X Clients under DESQview/X.

5.1 Real Mode Applications Development

In order to generate a real mode application for DOS a developer will follow the traditional steps shown on the right to produce an application.

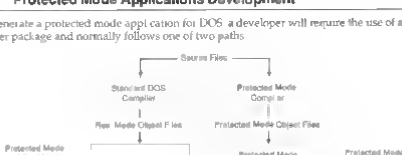
First all necessary source files are compiled using a standard DOS compiler to create real mode object files. Next a standard DOS linker is used to link those object files with real mode library modules to produce a runnable application.

Typically the compiler linker and library modules are supplied by a single manufacturer as a complete package.



5.2 Protected Mode Applications Development

To generate a protected mode application for DOS a developer will require the use of a DOS Extender package and normally follows one of two paths.



5.2.1 Using a Protected Mode Compiler and Linker

If a protected mode compiler is used it will generate protected mode object files. (Note that the words 'protected mode' in the name 'protected mode compiler' are referring to the kind of output the compiler generates, not the kind of program the compiler may be - it could actually be a real mode program or a protected mode program running under a DOS Extender!)

These protected mode object files are then linked with protected mode library files and DOS Extender modules to create a protected mode application that is runnable from DOS.

Typically the compiler linker and library modules are supplied by a single manufacturer as a complete package.

- If a 16-bit protected mode application is required then a 16-bit protected mode compiler linker and library modules must be used
- For a 32-bit protected mode application, a 32-bit protected mode compiler linker and library modules must be used

Note that a trend in protected mode compilers is to offer a DOS Extender as part of the compiler package obviating the need to choose and purchase a DOS Extender separately Previously there was not as big a selection of 16-bit and 32 bit protected mode development packages as there are today To address this situation many DOS Extender manufacturers supply an alternate route using standard DOS compilers

5.2.2 Using a Standard DOS Compiler

If a standard DOS compiler is used this will create real mode object files This may seem inconsistent however real mode is very similar to 16 bit protected mode code save for a few constraints Note that if generating code for a 32-bit environment using a standard (16-bit) DOS compiler will result in code that will not take advantage of the 32 bit architecture of the processor

At this point either a standard DOS linker may be used or a protected mode linker

Whatever linker is used it will typically link in real mode library routines and some protected mode modules as well in addition to the DOS Extender modules The real mode library routines are ones supplied by the standard DOS compiler manufacturer that do not violate protected mode guidelines and hence may be used in a protected mode environment Any library modules that do violate those guidelines are replaced by modules that have been rewritten by the DOS Extender manufacturers and are linked in as protected mode modules.

Sometimes it is necessary to run a conversion program after the linking stage to create the final protected mode program.

Note that a need for protected mode linkers has become apparent because many standard DOS linkers have certain limitations when creating protected mode programs (since they were not designed to produce these kinds of programs)

5.3 DOS Extenders

There are several DOS Extenders on the market Ones of particular note are

286 DOS Extenders (16-bit protected mode)	
DX16M	Rational Systems Inc Natick, Massachusetts USA
OS286	Ergo Computing, Inc Peabody Massachusetts USA
386 DOS Extenders (32-bit protected mode)	
386 DOS Extender	Phar Lap Software Cambridge Massachusetts USA
OS386	Ergo Computing Inc Peabody Massachusetts USA

Particular notes to be aware of when evaluating DOS Extenders are listed next

Compilers Compilers are not included by the DOS Extender manufacturer
Carefully check the compilers and linkers supported by the manufacturer
Different compiler/linker combinations can produce different maximum code sizes

Linkers	Sometimes linkers are supplied by the DOS Extender manufacturer
Debuggers	Debuggers may be supplied by the DOS Extender manufacturer or the compiler manufacturer Debuggers vary greatly in capabilities - some are symbolic some source level
Miscellaneous	Overlay technology may or may not be available Virtual Memory is usually an optional extra Costs vary greatly

As a final note Quarterdeck will supply a list of possible compilers and DOS Extender combinations with the CD variant DESQview/X Development Kits (see the product section for details)

5.4 X Client Development for DESQview/X

Developing or porting X Clients to the DESQview/X platform requires a developer to follow the general steps outlined in the previous section.

Depending on the size of the resultant X Client a developer will create either real mode 16-bit protected mode or 32 bit protected mode applications. Typically X Clients that are ported from another environment (usually Unix) will be implemented the easiest as a 32 bit protected mode application.

In order to create an X Client as opposed to a regular DOS or DOS Extended application the X Client object files are linked with Xlib and/or Toolkit function libraries in addition to the usual program libraries.

5.5 X Program Libraries

The program libraries and versions that will be supplied in the DESQview/X development package are as follows

Library	Real Mode	16-Bit Protected Mode	32 Bit Protected Mode
X11*	Y	Y	Y
OSF/Motif	N	Y	Y
Xol	N	Y	Y
Xview	N	Y	Y
XVT	Y	Y	Y

*The X11 library consists of the Xlib and Xt Intrinsics programming libraries

OSF/Motif Xol and Xview libraries will not be supplied for use by real mode applications due to the size of their libraries.

Other libraries in addition to the ones listed above may become available from Quarterdeck Office Systems

6 DESQview/X Products

There will be several products available in the DESQview/X suite. I system software, an end user product, development kits and a network software product.

All products have a planned availability date of first quarter 1991.

6.1 The DESQview/X End User Product

The end user product will enable a single user to implement a DESQview/X graphical environment system for running DOS and/or DOS-based X Clients. It will include the X Server product, a window manager and several graphical utilities such as an application manager.

Three versions will be initially available:

- DESQview/X and the DWM window manager
- DESQview/X and the OSF/Motif window manager*
- DESQview/X and an OPEN LOOK window manager*

*The DWM window manager will also be included with these products.

Additional window managers may also be supplied by Quarterdeck Office Systems.

6.2 The DESQview/X Development Products

The DESQview/X development environment consists of two classes of product: a base development product and additional toolkit libraries.

The base development kit consists of the following:

- DESQview/X End User Product and the X11 (Xlib and Xt Intrinsics) libraries

Hence, the base development product will include the X Server product, the DWM window manager and several graphical utilities in addition to the X11 libraries. This product allows a developer to create DOS-based X Clients using a third party compiler and for clients larger than 640K a DOS Extender.

Toolkits, which will allow programs to produce a specific look and feel, will be available in addition to the base product. Available toolkits will include:

- OSF/Motif - the OSF/Motif libraries and window manager
- Xol - the Xol libraries and window manager (OPEN LOOK)
- Xview - the Xview libraries and window manager (OPEN LOOK)
- XVT - the XVT libraries

Note that applications generated with the XVT Toolkit can also be used with Unix based OSF/Motif and Xol Toolkits or Windows, Presentation Manager and Macintosh libraries to produce applications that run under non-DESQview/X systems. These additional toolkits and libraries are supplied by companies other than Quarterdeck Office Systems.

6.3 The Network Software Product

The network software product enables a DESQview/X system to communicate with other X machines over a network.

Versions of the product will be available for the following networks:

- Ethernet (TCP/IP) using FTP software
- Novell (IPX)

Note that these packages provide DESQview/X with the capability to communicate with other DESQview/X and X Window machines over a network. They are not a substitute for and do not replace the standard network software that is required to form a network.

Disclaimer

It is noted that any products, features, or capabilities or availability dates described in this document are the current intentions of Quarterdeck Office Systems (20 August 1990) and in no way constitute a binding contract of any form and with any party whatsoever. The information presented in this document is for instructional purposes only. Everything described in this document is subject to change (including cancellation) and Quarterdeck Office Systems assumes no responsibility at all from any consequences incurred either directly or indirectly that depend on or are influenced by the information herein contained.

Trademarks

DESQview is a registered trademark of Quarterdeck Office Systems.
DESQview 386, *DESQview/X*, *DESQview API*, *QRAM*, *QEMM-50/60* and *QEMM-386* are trademarks of Quarterdeck Office Systems.
386 DOS Extender is either a trademark or registered trademark of Phar Lap Software
AutoCAD is a registered trademark of Autodesk, Inc.
Clipper is a registered trademark of Nantucket Corporation
GEM is a registered trademark of Digital Research Inc.
IGEM is either a trademark or registered trademark of Ravasoft Systems Inc.
Dbase is a registered trademark of Ashton-Tate Corporation
Dbase IV is a trademark of Ashton-Tate Corporation
DEC is a registered trademark of Digital Equipment Corporation
Intel 386 and *486* are trademarks of Intel Corporation
Lotus and *1-2-3* are registered trademarks of Lotus Development Corporation.
Macintosh is a registered trademark of Apple Computer Inc.
Microsoft and *MS-DOS* are registered trademarks of Microsoft Corporation
Novell is a trademark of Novell, Inc.
Open Look is a trademark of AT&T
OS/2® and *OS/386* are either trademarks or registered trademarks of Ergo Computing Inc.
OSF/Motif, *OSF* and *Open Software Foundation* are trademarks of the Open Software Foundation, Inc.
Paradox, *Paradox 386* and *Ansa* are either trademarks or registered trademarks of Borland International
PC-DOS, *Presentation Manager* and *OS/2* are trademarks of International Business Machines Corporation.
SunView is a trademark of Sun Microsystems Inc.
Unix is a registered trademark of AT&T in the US and other countries
Ventura Publisher is a trademark of Ventura Software Inc.
Windows is a trademark of Microsoft Corporation
WordPerfect is a registered trademark of WordPerfect Corporation.
X Window System is a trademark of the Massachusetts Institute of Technology

Copyright

This document is copyrighted and all rights reserved
This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent in writing, from Quarterdeck Office Systems

© 1990 Quarterdeck Office Systems All rights reserved

Credits

Document designed and written by Mark JC Radcliffe (QOS)
Document typography by Ventura Publisher

Quarterdeck Office Systems

150 Pico Boulevard, Santa Monica
CA 90405 USA
(213) 392-9851 Fax (213) 399-3802

Quarterdeck Office Systems Canada
Toronto, Ontario
(416) 516-1349

Quarterdeck International Limited
Dun Laoghaire Ireland
(01) 841-444

Quarterdeck Office Systems UK Limited
Chelmsford Essex
(0245) 496699